REMARKS

This is intended as a full and complete response to the Office Action dated November 29, 2006, having a shortened statutory period for response set to expire on February 28, 2007. Please reconsider the claims pending in the application for reasons discussed below.

Claims 1, 3-10, 12-19 and 21-26 are pending in the application. Claims 1, 3, 4, 7-10, 12, 13, 16-19, 21, 22, 24, and 26 remain pending following entry of this response. Claims 1, 7, 10, 16, 19, 24, and 26 have been amended. Claims 5, 6, 14, 23, and 25 have been cancelled. Applicants submit that the amendments do not introduce new matter.

Claim Rejections - 35 U.S.C. § 103

Claim 1 stands rejected under 35 U.S.C. § 103(a) as being unpatentable over *Krishnan* (US Patent 6,222,856 B1) in view of *Firth et al* (US 5,987,517). Applicants respectfully traverse this rejection.

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art, to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Third, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the third criteria.

The Examiner takes the position that *Krishnan* in view of *Firth*, discloses a computer-implemented method for providing asynchronous network communications between a client and a server that includes, in response to a request from the client, issuing a single, continuous mode operation to a socket, wherein the single, continuous

mode operation is selected from at least one of a single a single asynchronous accept operation and a single asynchronous receive operation. Applicants respectfully disagree.

As claimed, issuing the single asynchronous accept operation includes placing a single pending accept data structure on a pending queue, and for each of the plurality of incoming client connections, copying contents of the single pending accept data structure to a completed accept data structure queued on a accept completion queue, wherein the single pending accept data structure remains on the pending queue after the contents are copied. Claims 10 and 19 recite a similar limitation. Also as claimed, issuing the single asynchronous receive operation includes placing a single pending receive data structure on a pending queue, and for each completed client request, copying contents of the pending receive data structure to a completed receive data structure queued on a receive completion queue. Claims 10 and 19 recite a similar limitation. Applicants submit that *Krishnam*, in view of *Firth*, fails to teach, show, or suggest a method that includes issuing a single, continuous mode operation to a socket, wherein the single, continuous mode operation is selected from at least one of a single a single asynchronous accept operation and a single asynchronous receive operation, in the manner claimed.

Krishnan is directed to:

> a bandwidth throttling system that controls bandwidth on a per virtual service basis. The bandwidth throttling system determines the amount of bandwidth used by each virtual service and selectively throttles requests for a particular virtual service independently of others based upon the bandwidth usage for the particular virtual service.

Krishnan, 3:1-7. As part of the bandwidth throttling system, *Krishnan* describes a method for handling client requests at an ISP. The Examiner relies on elements of this method to suggest that *Krishnan* discloses the claimed step of issuing a single, continuous mode operation, as recited by claims 1, 10, and 19. Specifically, the Examiner suggests that:

> [*Krishnan* discloses] a continuous mode input operation (assigns an asynchronous thread context from the ATG library 80 to handle the

> request, col 8, In 6-8/ the ATO library also supports the I/O operating with respect to the network by providing functions to reads, writer [sic] and transmit files, col 8, In 10-13/ All the subsequent asynchronous I/O operations, col 8, In 20-21/ the ATQ library 80 enables asynchronous input and output operations, col 6, In 33-37) a listening socket (socket are stored in the ATQ Context, col8, In 13-15/ the ATO uses the context value, col 8, In 53-55).

Office Action, p. 2. As is known, a "context" refers to a set of resources (memory, registers), and the state of those resources, and a context switch is the process of storing and restoring the state (context) of a CPU such that multiple threads can share a single CPU resource. However, that the "ISP server assigns an asynchronous thread context (ATQ Context) form the ATQ library to handle the incoming request," *Krishnan*, 8:7-9, fails to disclose the single, continuous mode operation, as claimed. By its own terms, this passages discloses assigning a context (a set of resources) to handle an including request.

More specifically, nothing in this material demonstrates that the "ATQ Library[1]" discussed in *Krishnan* issues, a continuous mode operation to a socket, selected from a single a single asynchronous accept operation or a single asynchronous receive operation, in the manner claimed. Instead, it discloses that a conventional thread library may be used to assign a thread context to process an incoming request, i.e., a thread context to use for "functions to read, write, and transmit data files over the network connection using socket capability." Krishnan, 8:11-13.

After Applicants appealed a rejection in this matter based on a combination of Applicants "admitted prior art" and *Firth*, the Examiner has reopened prosecution and, essentially, swapped out a rejection based on a description of conventional socket processing techniques disclosed in Application, p. 1-5 with a rejection based on a description of a conventional thread processing library discussed in *Krishnan*, 8:6-8, 8:10-13, 8:53-55.

---

[1] Applicants note that the Examiner refers separately to an "ATQ library," an "ATG library," a "QAT" library, and an "ATO library." Applicants assume that in each case the Examiner intended to refer to the "ATQ library" discussed in *Krishnan*, column 8.

Further, the Examiner concedes that "*Krishnan* does not explicit [sic] teach the single asynchronous operation[2]," which makes sense as *Krishnan* merely makes use of a conventional thread library as part of a system for throttling bandwidth on a "per virtual service basis." Nevertheless, the Examiner continues to rely on *Firth* as teaching this element. However, as demonstrated in Applicants Appeal brief in this matter:

> The material cited from *Firth* (and *Firth* generally) fails to disclose techniques for managing socket-based communications. Rather, *Firth* is directed to an API of functions that provide an abstraction of data-communication techniques. The abstraction allows developers to create high-level applications without having to understand or manage the details of an underlying data communication mechanism. That is, the abstraction provided by the "Internet API" allows developers to create software applications without having to understand how a particular socket-based communication may occur. For example, *Firth* provides;
>
> > In the preferred embodiment of the present invention, calls to two of the reentrant Internet API functions (e.g., InternetOpen(. . .),InternetConnect(. . . ,FTP, . . .), which will be explained in detail below) will initialized an Internet session, establish a connection, and manage all the underlying details including the FTP protocol, the communication facilities required (e.g., a socket connection), ... The Internet application program does not have to include source code to establish an Internet connection, handle the FTP protocol, the communications facilities, or the underlying protocols. All of these details are abstracted in the Internet API and are hidden from or transparent to the application program.
>
> *Firth*, 3:37-52. As the emphasized passage makes clear, *Firth* discloses techniques that allow an application developer to compose an application without an understanding of "the underlying protocols." Not surprisingly, therefore, *Firth* fails to teach or suggest mechanisms for providing the

[2] Previous rejections in this matter provided "[Admitted Prior art] does not explicit teach the single asynchronous operation."

lower-level asynchronous network communications, in the manner claimed.

In asserting that "*Firth* teaches single asynchronous operation," [Present Office Action, p. 3], the Examiner cites passages directed to aspects of the "Internet API" disclosed in Firth. These passages highlight that *Firth* is directed to functions provided by an "Internet API," and does not, in fact, disclose limitations of a method for providing asynchronous network communications between a client and a server, as claimed by Applicants

> For example, Examiner cites to *Firth*, 16:23-27, which provides:
>
> As was just described, a single call to the InternetOpen ( ) function from the internet API provides a client application with the ability to select the type of Internet access, select a proxy for a first level of security, ... [or perform other various actions]."

As disclosed in *Firth*, the InternetOpen ( ) function is used to '"initialize the application's use of the reentrant Internet API functions. *Firth*, 9:35-38. In other words, an application calls the InternetOpen function in order to use other functions provided by the "Internet API" disclosed by *Firth*. For example, *Firth* describes how the InternetOpen () function must be called prior to a call to another "Internet API," function, InternetConnect():

> As an example, InternetConnect(), which is a first level dependent function, cannot be called until InternetOpen () (an independent function) is first called and returns a valid Internet handle, which is a required argument for InternetConnect() call. If an application desires to find the first file located during an FTP session with a connection to the Internet, InternetOpen () is called and the Internet handle that is returned is used as an argument for a call to InternetConnect(. . . ,FTP, . . .) to establish an FTP application protocol session. Finally, the Internet handle returned from InternetConnect() is used as an argument in a call to FtpFindFirstFile(). Other dependent functions use handles from the next higher level in a similar manner.

> *Firth*, 11:48-61. Applicants submit that initializing the "Internet API" of *Firth* using the InternetOpen() function of the "Internet API" fails to disclose "issuing a single, continuous mode operation" to a socket using a "single asynchronous accept operation" or a "single asynchronous receive operation," in the manner claimed. Rather, the IntenetOpen() function allows an application developer to make function calls to other functions of the "Internet API" disclosed in *Firth*. These functions operate independently from the mechanism used to manage socket-based communications. In fact, this is the whole point of the "Internet API": to provide an API where the details of a socket-based communication are "abstracted in the Internet API and are hidden from or transparent to the application program." *Firth*, 3:50-52.

See *Appeal Brief* dated April 24, 2006. Applicants submit that, as with the previous rejections, the reliance on the InternetOpen() function to demonstrate that *Firth* discloses "single asynchronous operation" in the manner suggested by the Examiner is misplaced.

For all the foregoing reasons, therefore, Applicants submit that independent claims 1, 10, and 19 are allowable. Accordingly, Applicants request withdrawal of this rejection and the allowance of claims 1, 10 and 19.

Claims 4, 7, 8, 10, 13, 16, 17, 19, 21, 23, 26 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Krishnan* in view of *Firth et al* (US 5,987,517), as applied to claim 1 above, and further in view of APA (Admitted Prior Art).

Claims 4, 7, 8, 10, 13, 16, 17, 19, 21, 23, 26 depend from one of claims 1, 10, or 19. As Applicants believe the above remarks demonstrate that *Krishnan* in view of *Firth* does not render the independent claims obvious, Applicants believe that these dependent claims are allowable over *Krishnan* in view of *Firth* and further in view of *APA*.

Claims 3, 12 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Krishnan* (US Patent 6,222,856 B1) in view of *Firth et al* (US 5,987,517), as applied to claim 1 above, and further in view of Shah et al (US Patent 6, 175, 879 B1).

Claims 3 and 12 depend from claims 1 and 10, respectively. As Applicants believe the above remarks demonstrate that *Krishnan* in view of *Firth* does not render the independent claims obvious, Applicants believe that these dependent claims are allowable over *Krishnan* in view of *Firth* and further in view of *Shah*.

Claims 9, 18, and 22 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Krishnan* (US Patent 6,222,856 B1) in view of *Firth et al* (US 5,987,517), as applied to claim 1 above, and further in view of APA (Admitted Prior Art) and further in view of *Joh* (US Patent 6,717,954 B1).

Claims 9, 18, and 22 claims depend from claims 1, 10, and 19, respectively. As Applicants believe the above remarks demonstrate that *Krishnan* in view of *Firth* fails to render the independent claims obvious, Applicants believe that these dependent claims are allowable over *Krishnan* in view of *Firth* and further in view of *Joh*.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicant(s)